MYWORLD: AN AGENT-ORIENTED TESTBED FOR DISTRIBUTED ARTIFICIAL INTELLIGENCE

Michael Wooldridge[*]
Department of Computing
Manchester Metropolitan University
Chester Street, Manchester M1 5GD
United Kingdom

Didier Vandekerckhove
Laboratoire D'Intelligence Artificielle
Université de Savoie
73376 Le Bourget-du-Lac CEDEX
France

ABSTRACT

Agent-Oriented Programming (AOP) is a new programming paradigm which proposes that mentalistic notions (such as belief, intention, commitment and so on) are useful abstraction mechanisms for describing complex, intelligent agents, and that they may therefore be used as a basis for programming such agents [17]. In this paper, we describe a Distributed Artificial Intelligence testbed called MYWORLD, in which individual agents are defined in terms of such notions. At run-time, a MYWORLD system contains a simulated *world*, containing a number of agents, (which are able to act in the world, for example by moving or communicating with other agents), and possibly other objects, with properties defined by the user. The organisation of MYWORLD and the agent-oriented programming language used are described in detail. A number of cooperation experiments performed with MYWORLD are also discussed. We conclude by outlining our plans for the next refinement of MYWORLD.

[*] **e-mail:** mikew@sun.com.mmu.ac.uk, **tel:** (+44 61) 247 1531, **fax:** (+44 61) 247 1483.

1 INTRODUCTION

As a research discipline, Distributed Artificial Intelligence (DAI) is less than fifteen years old; even by the standards of AI, this makes it something of an infant. In all sciences at such an early stage of evolution, *experimentation* must play a key role, developing the concepts that formalists ultimately transform into mathematical theories. In DAI, many software tools have been developed to support experimentation (e.g., [1, 11, 12, 8, 21, 5, 2, 10]). In parallel with this work, there has been an ongoing debate in (D)AI on the question of how best to construct the agents which make up a (D)AI system. One proposal, due to Shoham [17], is to program agents in terms of *mentalistic* notions: belief, intention, commitment, and so on. The result of such a treatment is a paradigm known as *agent-oriented programming*. In this paper, we describe MYWORLD, an *agent-oriented programming testbed for DAI*. MYWORLD is a software platform for DAI experimentation in which agents are programmed along the lines proposed by Shoham. Thus the state of an agent in MYWORLD may be characterised as a set of *beliefs*, a set of *intentions*, a set of rules governing the generation and revision of beliefs and intentions, and a set of rules defining how to achieve intentions. Agents in MYWORLD exist in an environment which includes a simple notion of space; they can move through this space, performing actions, and interacting with other agents. The MYWORLD user has the ability to create any number of 'worlds' with different characteristics, in which experiments can be run.

We begin in §2 by describing the background to our work, and in particular Shoham's proposal; in §3, we describe MYWORLD in more detail. In §4, we outline some experiments we have performed with MYWORLD, and in §5, we point to future work areas.

2 BACKGROUND

As we observed above, experimentation is a crucial aspect of DAI research. To support experimentation, a number of software testbeds, languages, and other platforms for DAI have been developed and reported in the literature (e.g., [1, 11, 7, 12, 8, 21, 5, 2, 10]). Some of these testbeds have been 'dedicated', that is, constructed to support experimentation in just one problem domain. The best example of such a platform the DVMT, (see, e.g., [7]), which allows a user to simulate distributed sensing problems. Other testbeds have offered more generality:

- MACE, probably the best-known DAI software, is a LISP-based 'instrumented testbed for building a wide range of experimental [DAI] systems at different levels of granularity' [11, p119];
- MCS/IPEM is a PROLOG-based testbed, which is novel in that agents have virtual access to a sophisticated non-linear planner [5];

- Concurrent METATEM is a DAI programming language in which each agent is defined by expressing its desired behaviour as a temporal logic theory, which is directly executed to produce the agent's actual behaviour [10];
- ARCHON is not so much a single piece of software as a set of related tools and techniques for building DAI systems, which have the distinction of having been applied to a complex real-world problem (electricity distribution management) [19];
- the Procedural Reasoning System (PRS) is a general agent architecture which attempts to marry principles from reactive and deliberative systems; it has been used in a number of DAI experiments [13, 12]; the theoretical properties of the PRS are investigated in [15];
- finally, MICE is a LISP-based DAI testbed which has been used particularly for predator-prey experiments [8].

Additionally, a number of Actor-style languages have been developed [1, 21, 2]: 'pure' Actor languages have an elegant semantics, and remain highly influential in programming language design [1]; other languages such as ABCL [21] and MAGES [2] have borrowed many concepts from the original Actors model.

All of the above systems have distinctive and individual features, and each has properties which make it well suited to some particular domains; any critical comparison is therefore likely to be somewhat superficial. However, one point worth noticing is that most of the systems mentioned above rely on an *informal* notion of *agency*[1]. Recently, however, Yoav Shoham proposed a programming paradigm called *agent-oriented programming* (AOP), which relies on a more formal notion of agency [17]. Crudely, his idea was that *mentalistic* notions (such as belief, commitment, intention, etc.), are useful *abstraction tools* for describing complex systems[2]. Shoham proposed that mentalistic concepts might therefore be useful for *programming* complex systems. He went on to define a logic containing modalities for belief and commitment, in which he hoped to express the properties of agents. The ultimate aim is to 'compile' an agent specification expressed in such a logic into an efficient automata-like machine, (in the spirit of [16]), though Shoham expresses some doubts about this process [17, p12]. In the meantime, an interpreted language called AGENT0 has been developed, in which agents are programmed with simple versions of the formal logical notions.

Whether or not the specifics of Shoham's proposal are ultimately viable, it seems that the general idea — that of programming agents in terms of mentalistic constructs — is an exciting one, that deserves evaluation. AOP seems particularly relevant to DAI, a discipline in which one must address the question 'what is an agent?'

[1]The exceptions are the pure actor languages (which have an elegant, rigorously defined semantics), and Concurrent METATEM, (which has a well-motivated logical foundation).

[2]This idea is called the *intentional stance* [4]. Similar ideas, involving the development of *knowledge-based protocols* have been investigated by the distributed systems community [9].

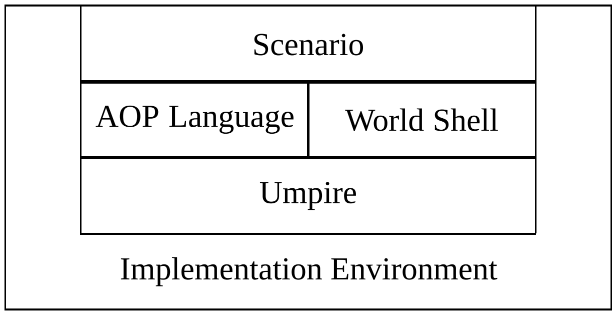

Figure 1: MYWORLD System Organisation

before one can begin. We therefore decided to build a DAI testbed called MYWORLD, in which agents are programmed along the lines that Shoham proposes.

3 MYWORLD

In this section, we describe MYWORLD in as much detail as space allows. We begin with an overview of the system, go on to describe its components in more detail, and conclude with a comment on the current implementation.

3.1 MYWORLD System Structure

At run-time, a MYWORLD system has four components:

- an *umpire*, or *world manager*, which has top-level control of the system;
- an AOP language, for programming agents;
- a *world shell*, which defines the characteristics of a particular 'world', (e.g., the environmental constraints that exist in that world);
- a *scenario*, which represents a particular experiment.

The organisation of these components is illustrated in Figure 1.

The Umpire

The umpire is that part of MYWORLD that has overall, top-level control of the system. Amongst other things, it is responsible for scheduling agent execution, monitoring the user interface, and responding to requests from the latter. The umpire is the 'generic' part of MYWORLD, and was designed to allow as much flexibility and generality as possible. However, it was found necessary to build certain basic principles into it:

- *Agents and other objects.* There are two types of entities in any MYWORLD system: *agents*, (which are in some sense 'active'), and *objects* (which are inanimate). Agents are characterised by two sets of attributes: *external* (which relate agents to the environment they occupy), and *internal* (e.g., the 'cognitive state' of the agent). Objects have no internal structure; they only have external attributes.

- *Space.* Any MYWORLD scenario takes place in a grid (typically something like 150×150 in size). All MYWORLD entities have at least one type of external attribute in common: a location in the grid.

- *Action.* Agents are also distinguished from objects in that they have the ability to *act*. Actions may be *physical*, (affecting the external attributes of one or more entities), or *cognitive*, (affecting the internal attributes of the actor).

- *Events.* Agents can only ever *attempt* to perform an action; in reality, actions (and particularly physical actions) may *fail*. The success or failure of an action is in the hands of the umpire, which follows the rules determined by the world shell. Whether an action succeeds or fails, it always generates an *event*, which describes the effect of the action on the world.

- *Time.* Time in MYWORLD is measured in scheduler cycles. At each cycle, every agent gets to execute one scheduler cycle; agent execution is therefore only pseudo-concurrent.

The AOP Language

The basic principle of AOP is that agents are programmed in terms of mentalistic notions such as belief, intention, commitment, and so on. However, there is some debate as to exactly which of these modalities are the most basic and/or appropriate. Shoham chooses just two modalities: belief (with a semantics defined via possible worlds, in the standard modal logical sense), and 'commitment', (with a semantics also defined via possible worlds) [17]. However, other authors have argued for different modalities: Cohen and Levesque suggest that beliefs and goals alone are sufficient to define other mental states such as intention [3]; Rao *et al.* have argued for a belief-desire-intention (BDI) agent architecture [14]. For MYWORLD, we have chosen just two modalities: belief and intention. Our decision to choose these was based on the observation that they could be given a simple and intuitive operational semantics, and that an intermediate 'desire' modality could always be simulated using beliefs and intentions, and could therefore be dispensed with.

An agent in MYWORLD can be characterised as a 6-tuple:

$$\langle \mathcal{B}, \mathcal{I}, \mathcal{BR}, \mathcal{IR}, \mathcal{SR}, \mathcal{A} \rangle$$

where:

- $\mathcal{B}$ is a set of *beliefs*;
- $\mathcal{I}$ is a set of *intentions*;
- $\mathcal{BR}$ is a set of *belief rules*;
- $\mathcal{IR}$ is a set of *intention adoption rules*;
- $\mathcal{SR}$ is a set of *strategy rules*; and
- $\mathcal{A}$ is a set of *actions*.

An agent's beliefs constitute that agent's information about the world it occupies (in more traditional AI terms, an agent's beliefs are its 'knowledge'); intentions represent desires that the agent will attempt to bring about; the belief rules define how new beliefs are generated from old ones; the intention adoption rules (IARs) determine how new intentions are generated; the strategy rules define methods for achieving intentions; and finally, the action set represents those actions that the agent can perform. These components, and the way that they interact to generate the behaviour of the agent, will now be described in more detail.

In the current implementation of MYWORLD, (see §3.2), an agent's beliefs are a set of ground atoms of first-order predicate logic (cf. PROLOG facts). An agent's belief set represents that agent's 'knowledge', which typically relates to the environment it occupies. A belief rule is a rule that can be applied to a belief set in order to generate new beliefs; in the current implementation, such a rule can be considered to be a first-order formula of the form

$$\forall \overline{x} \cdot \phi(\overline{x}) \Rightarrow \psi(\overline{y})$$

where $\overline{x} = x_1, \ldots, x_m$ and $\overline{y} = y_1, \ldots, y_n$ are tuples of variables, such that

$$\{x_1, \ldots, x_m\} \subseteq \{y_1, \ldots, y_n\}$$

and $\phi(\overline{x})$ and $\psi(\overline{y})$ are conjunctions of predicates over the variables $\overline{x}$ and $\overline{y}$, respectively. Given a belief set, the way in which such a rule is applied to generate new beliefs is obvious. In the current implementation, an agent's belief rules are applied wherever possible; there are obviously potential efficiency problems here, when dealing with large belief sets and large numbers of belief rules, which will be addressed in future implementations (see §5).

Belief sets are not fixed: they may change over time, by new beliefs being added and old beliefs being removed. New beliefs arise from three sources:

- from inferences made via belief rules, in the way described above;
- from perceiving the world; and
- from performing non-logical 'cognitive', or 'private' actions.

Perception in MyWorld is simulated by the umpire: each agent has an associated external attribute called its perception rating, which indicates how ‘far’ that agent can ‘see’. On the basis of this value, and the agent’s position in the MyWorld space, the umpire generates a set of beliefs describing those things which are visible to the agent; this belief set is then made available to the agent.

Cognitive, or private actions, correspond to an agent exploiting its computational resources. For example, imagine an agent with access to a database; consulting this database would be an action performed ‘inside’ the agent, and would not be visible to other agents; hence the term ‘cognitive’ action. The result of such an action would be some information, which appears in the form of some new beliefs.

Finally, old beliefs may be removed from the belief set. In the current implementation, we remove certain old beliefs in order to improve the efficiency of the system; we have not seriously addressed the problem of belief revision, though we intend to in future work.

An *intention* is a triple

$$\langle \phi(\overline{a}), \psi(\overline{b}), n \rangle$$

where

- $\phi(\overline{a})$ is a conjunction of predicates over the constants $\overline{a}$ called the *goal* of the intention;
- $\psi(\overline{b})$ is a conjunction of predicates over the constants $\overline{b}$ called the *motivation* of the intention; and
- $n \in \mathbb{N}$ is a natural number called the *rating* or *priority* of the intention.

The idea is that the goal describes the ‘purpose’ of the intention — if it ever becomes satisfied, then the intention is fulfilled. The motivation describes what must be believed in order for the intention to be maintained — if the motivation ever becomes false, then there is no point in maintaining the intention, and it is dropped. So an intention will be maintained until either its goal is believed to be satisfied or its motivation is no longer present[3]. The *rating* of an intention indicates its priority; the higher the rating, the more important the intention. The highest rated intention is called the *current* intention, and will guide the actions of the agent, in a way that we describe below.

We now consider an example, to illustrate the working of intentions. In our experiments with MyWorld, (see §4), agents must ‘eat’ resources that are distributed at random throughout the MyWorld space. Agents can only eat a resource if they occupy the same location. The general behaviour of an agent thus consists of repeating the steps: (i) locate a resource; (ii) move to the resource; and (iii) eat it. When an agent locates a resource, say at grid location $(100, 20)$, it generates an intention such as the following:

$$\langle \textit{Me-In}(100, 20),\ \textit{Resource}(100, 20),\ 7 \rangle$$

[3]This is very close to what Cohen and Levesque call a *persistent relativised goal* [3].

The goal of this intention is $\textit{Me-In}(100, 20)$, which will be satisfied if the agent believes it is in location $(100, 20)$. The motivation for the intention is $\textit{Resource}(100, 20)$, which will be satisfied while there is a resource at location $(100, 20)$. (The rating of the intention is 7, which is about mid-range for our experiments.) So, the intention will be maintained until either the agent believes itself to be in location $(100, 20)$, or the agent no longer believes that there is a resource at $(100, 20)$.

New intentions are generated from intention adoption rules (IARs). An IAR is a pair, containing:

- an *adoption condition* (a predicate containing variables); and
- an *intention skeleton* (an intention containing variables).

The idea is that on every cycle, the agent tries to match the adoption condition of each IAR with its beliefs; if it succeeds, then it instantiates the variables of the corresponding intention skeleton, and adds the resulting intention to its set of intentions. Additionally, on every cycle, the intention set is pruned by removing those intentions that are either satisfied, or whose motivation is no longer present.

Let us consider an example, related to that above. Agents in our experiments are given an intention adoption rule with the following general form:

$$\langle \overbrace{\textit{Resource}(x,y)}^{\textit{adoption condition}},\ \underbrace{\langle \textit{Me-In}(x,y),\ \textit{Resource}(x,y),\ 7\rangle}_{\textit{intention skeleton}}\ \rangle$$

Suppose an agent then had the belief $\textit{Resource}(100, 20)$, then it would attempt to match the condition part of this IAR with its beliefs and would succeed, with unification resulting in the following binding: $\{x \mapsto 100, y \mapsto 20\}$. After applying this binding to the intention skeleton, the following new intention would be generated

$$\langle \textit{Me-In}(100, 20),\ \textit{Resource}(100, 20),\ 7\rangle.$$

Thus whenever an agent sees a resource, it would generate an intention to go to it[4].

Intentions are linked to actions by a set of *strategy rules*. Strategy rules describe how to achieve intentions. A strategy rule is a pair, consisting of:

- a *predicate symbol*, corresponding to the goal of an intention;
- a *strategy function*, which takes as its sole argument the state of the agent, and returns as its sole result a member of $\mathcal{A}$, (recall that $\mathcal{A}$ is the set of actions that the agent can perform), representing the action that the agent has chosen to perform.

[4]Note that: (i) in the actual implementation, there is a mechanism for choosing between resources, so as to pick the most desirable; and (ii) although in this example the adoption condition and motivation are the same, this need not be the case.

To illustrate the working of strategy rules, we consider an example, again based on the idea of an agent moving to a resource. Agents in the experiment are equipped with a strategy rule such as the following:

$$\langle \textit{Me-In}, \textit{go-to} \rangle$$

Me-In is a predicate symbol that describes the goal of an intention; *go-to* is a function that, when evaluated, looks at the agent's internal state to see exactly where the agent wants to go, and returns an action describing the appropriate movement for the agent (e.g., $\textit{Move}(N)$, $\textit{Move}(S)$, $\textit{Move}(E)$, $\ldots$). On every scheduler cycle, an agent's current intentions are checked to find the highest rated one; the goal predicate of this intention is then matched against the strategy rule, to find an appropriate strategy function, which is then evaluated in order to find the agent's chosen action.

Strategy rules therefore contain information about achieving intentions; they thus resemble knowledge areas in the PRS [13]. Strategy functions are the closest that agents in MYWORLD come to doing any planning; strategy functions could be thought of as crude procedural plans. Note that there must be exactly one strategy rule for every possible intention goal predicate (in the terminology of [20], the strategy rule set must be *strongly complete*).

Let us now summarise the behaviour of an agent on a single scheduler cycle:

1. update beliefs by adding any new beliefs and applying belief rules exhaustively;
2. update intentions by:
   - removing those that are no longer applicable; and
   - finding and adding those that have now become applicable;
3. find the highest rated intention and match it against strategy rules in order to find a strategy function;
4. evaluate the strategy function in order to find some action;
5. return the action that results.

World Shell

The world shell is that part of MYWORLD that defines the environmental characteristics of a particular 'world' in which experiments may be performed. For example, the world shell defines what actions may be performed in an experiment, when such actions are legal, and what the results of an action performed in some particular situation are. Additionally, it defines what entities may appear in the world, and what their properties are. At the time of writing, we have implemented just one world shell, for the experiments described in §4; in future, we hope to construct a library of world shells for domains as the tile world (see, e.g., [12]).

Scenario

A scenario in MYWORLD represents a specific experiment carried out. It describes the initial locations and properties of all the entities that appear in the experiment.

3.2 IMPLEMENTATION ASPECTS

MYWORLD has been implemented in the language POP-11; the implementation contains about 3000 lines (85k) of code; a typical scenario involves something like 1000 lines (30k) of code. On average, our experiments have involved approximately ten agents, each with about thirty IARs. On a SPARC-1, a single cycle in such a system takes about two seconds; we believe that this performance can be improved significantly.

The implementation includes an X-windows interface, with a graphical representation of the world, and provides simple facilities for debugging and tracing agents. For example, if the user clicks on the graphical representation of any agent, another window will be created showing that agent's current beliefs and intentions, what action it just performed, and possibly scenario specific information.

4 EXPERIMENTS

We have conducted a number of experiments with MYWORLD; most of these have been based on recreations of limited parts of the EOS project [6][5]. In this scenario, the world is populated by just two types of entities: agents and resources. The goal of each agents is to 'eat' as many resources as possible; eating a resource is achieved by an agent moving to a resource and executing an *eat* action. However, resources have an associated 'cost', which represents the number of agents required to obtain the resource; for example, if the cost is 3, then 3 agents must simultaneously execute an eat action to obtain the resource. There are a limited number of resources of cost 1 in each scenario, so agents are ultimately forced to act cooperatively if they wish to eat. Cooperative activity is achieved through a version of the contract net protocol ([18]): if an agent sees a resource of cost greater than 1, then it locates some 'free' agents and 'advertises' the resource to them. Agents may then 'bid' for the resource; the advertiser subsequently 'awards' the resource to the appropriate number of bidders, which then cooperatively obtain the resource under instruction from the leader.

[5]We stress that it has not been our aim to recreate the results of this project in its entirety, nor to produce original results in the domain of the project (the emergence of social structure in prehistoric societies). Rather, we chose the domain because it was both simple enough to manage for a prototype system, and yet rich enough to allow significant experimentation with varying parameters.

5 CONCLUDING REMARKS

MyWorld is very much an exploratory system. While we have used it to carry out experiments of meaningful size and complexity, there are many desirable features which it lacks, and many undesirable features which it has. The next generation of MyWorld will differ from the current implementation in the following key respects: improved, enriched language for belief rules and IARs; better motivated communication model; more efficient algorithms for updating beliefs and intentions; improved, enriched user interface; more facilities for defining experiments and scenarios; and intelligent explanation tools. Additionally, we are working on a formal specification of MyWorld and its successor, and trying to incorporate some insights from our more theoretical work on multi-agent systems [10, 20].

REFERENCES

[1] G. Agha. *ACTORS: A Model of Concurrent Computation in Distributed Systems.* The MIT Press, 1986.

[2] T. Bouron. A multi-agent simulation system with high-level structure implemented in an actor language. In Y. Demazeau and J. P. Muller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW '90)*. Elsevier/North Holland, 1991.

[3] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[4] D. C. Dennett. *The Intentional Stance*. Bradford Books/MIT Press, 1987.

[5] J. Doran, H. Carvajal, Y. J. Choo, and Y. Li. The MCS multi-agent testbed: Developments and experiments. In S. M. Deen, editor, *CKBS '90 — Proceedings of the International Working Conference on Cooperating Knowledge Based Systems.* Springer-Verlag, 1991.

[6] J. Doran, M. Palmer, N. Gilbert, and P. Mellars. The EOS project. In *Simulating Societies '92: Pre-proceedings of the 1992 International Symposium on Approaches to Simulating Social Phenomena and Social Processes*, Department of Sociology, University of Surrey, 1992.

[7] E. Durfee. *Coordination of Distributed Problem Solvers.* Kluwer Academic Press, 1988.

[8] E. Durfee and T. Montgomery. MICE: A flexible testbed for intelligent coordination experiments. In *Proceedings of the 1989 International Workshop on Distributed Artificial Intelligence*, 1989.

[9] R. Fagin, J. Y. Halpern, and M. Y. Vardi. What can machines know? on the properties of knowledge in distributed systems. *Journal of the ACM*, 39(2):328–376, 1992.

[10] M. Fisher and M. Wooldridge. Executable temporal logic for distributed A.I. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, Hidden Valley, PA, May 1993.

[11] L. Gasser, C. Braganza, and N. Hermann. MACE: A flexible testbed for distributed AI research. In M. Huhns, editor, *Distributed Artificial Intelligence*. Pitman/Morgan Kaufmann, 1987.

[12] M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI '89)*, Detroit, MI, 1989.

[13] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI '87)*, Seattle, WA, 1987.

[14] A. S. Rao and M. P. Georgeff. Modeling agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*. Morgan Kaufmann Publishers, Inc., April 1991.

[15] A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*, pages 439–449, 1992.

[16] S. Rosenschein and L. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*. Morgan Kaufmann Publishers, Inc., 1986.

[17] Y. Shoham. Agent-oriented programming. Technical Report STAN–CS–1335–90, Department of Computer Science, Stanford University, 1990.

[18] R. G. Smith. The contract net protocol. *IEEE Transactions on Computers*, C-29(12), 1980.

[19] T. Wittig, editor. *ARCHON: An Architecture for Multi-Agent Systems*. Ellis Horwood, 1992.

[20] M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992.

[21] A. Yonezawa, editor. *ABCL — An Object-Oriented Concurrent System*. The MIT Press, 1990.